Oracle Direct Seminar

ORACLE®

基礎から学ぶXML特集
- Oracleで始めるXML -

日本オラクル株式会社

Oracle Direct

Oracle Direct Seminar

アジェンダ

- Oracle Database に XML 文書を格納する
- Oracle Database で XPath を使う
- Oracle Database で XQuery を使う
- Oracle Database でリレーショナルデータをXML形式で参照する
- Oracle Database でXMLデータをリレーショナル形式で参照する
- "2択"にチャレンジ

事前準備

- OTN から Oracle Database のバイナリをダウンロード
- Oracle Database をインストール
- DBCA（Database Configuration Assistant）を利用してデータベースを作成する
  - データベース構成タイプとして「汎用（General Purpose）」を選択します

［参考］

『意外と簡単!? Oracle Database 11g Release 2 データベース構築から運用まで』

http://www.oracle.com/technology/global/jp/columns/easy/oracle11gr2/windows/pdf/SelfStudy_Win_11.2_01.pdf

Oracle Database に XML 文書を格納する

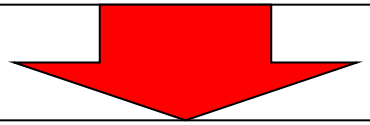
Oracle Database と XML文書

- Oracle Database に XML 文書を格納するには？
  - VARCHAR2 型を用いる
  - CLOB 型を用いる
  - **XMLType 型を用いる**（Oracle9i Database R1～）

XMLType 型の簡単な使い方

- XMLType コンストラクタ
  - VARCHAR2、CLOB、BLOB、BFILE 型からXMLType型への変換を行う（＝XMLTypeインスタンスを作成する）
- 使用例

```
CREATE TABLE testxml (xml XMLTYPE);
INSERT INTO testxml VALUES (XMLTYPE('<a>dummy</a>'));
COMMIT;
```



```
SQL> SELECT xml FROM testxml;
XML
--------------------------------------------------------------------------------
<a>dummy</a>
```

問題文の引用元
http://www.xmlmaster.org/sample/sample_basic.html#02

サンプル試験問題
XMLマスター：ベーシック 試験（1）

次のXML文書のうち、妥当なXML文書はどれでしょうか。正しいものを選択してください。

A.

```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1,elmnt2?)+>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
</rtelmnt>
```

B.

```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1,elmnt2?)+>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
</rtelmnt>
```

C.

```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
</rtelmnt>
```

D.

```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
</rtelmnt>
```

問題文の引用元
http://www.xmlmaster.org/sample/sample_basic.html#02

サンプル試験問題
XMLマスター：ベーシック 試験（1）

次のXML文書のうち、妥当なXML文書はどれでしょうか。正しいものを選択してください。

A.

```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1,elmnt2?)+>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
</rtelmnt>
```

B.

```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1,elmnt2?)+>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
</rtelmnt>
```

C.

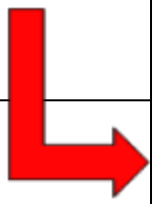
```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
   <elmnt1></elmnt1>
   <elmnt2></elmnt2>
</rtelmnt>
```

D.

```
<?xml version="1.0"?>
<!DOCTYPE rtelmnt[
<!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
<!ELEMENT elmnt1 (#PCDATA)>
<!ELEMENT elmnt2 (#PCDATA)>
]>
<rtelmnt>
</rtelmnt>
```

選択肢 C の XML 文書を、XMLTYPE型としてコンストラクトしようとすると・・・

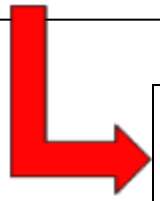
```
1  DECLARE
2    tempxml XMLTYPE;
3  BEGIN
4    tempxml := XMLTYPE('<?xml version="1.0"?>
5  <!DOCTYPE rtelmnt[
6  <!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
7  <!ELEMENT elmnt1 (#PCDATA)>
8  <!ELEMENT elmnt2 (#PCDATA)>
9  ]>
10 <rtelmnt>
11    <elmnt1></elmnt1>
12    <elmnt2></elmnt2>
13    <elmnt1></elmnt1>
14    <elmnt2></elmnt2>
15 </rtelmnt>');
16 END;
17 /
```



```
DECLARE
*
行1でエラーが発生しました。:
ORA-31011: XML解析に失敗しました
ORA-19202: XML処理
LPX-00103: Warning: ドキュメント構造がDTDと一致しません。
Error at line 10
中にエラーが発生しました ORA-06512:
"SYS.XMLTYPE", 行310
ORA-06512: 行4
```

選択肢 C の XML 文書に変更を加える(1)

```
1  DECLARE
2    tempxml XMLTYPE;
3  BEGIN
4    tempxml := XMLTYPE('<?xml version="1.0"?>
5  <!DOCTYPE rtelmnt[
6  <!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
7  <!ELEMENT elmnt1 (#PCDATA)>
8  <!ELEMENT elmnt2 (#PCDATA)>
9  ]>
10 <rtelmnt>
11    <elmnt1></elmnt1>
12    <elmnt2></elmnt2>
13 </rtelmnt>');
14 END;
15 /
```



PL/SQLプロシージャが正常に完了しました。

選択肢 C の XML 文書に変更を加える(2)

```
1  DECLARE
2    tempxml XMLTYPE;
3  BEGIN
4    tempxml := XMLTYPE('<?xml version="1.0"?>
5  <!DOCTYPE rtelmnt[
6  <!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
7  <!ELEMENT elmnt1 (#PCDATA)>
8  <!ELEMENT elmnt2 (#PCDATA)>
9  ]>
10 <rtelmnt>
11    <elmnt1></elmnt1>
12    <elmnt1></elmnt1>
13    <elmnt1></elmnt1>
14    <elmnt1></elmnt1>
15    <elmnt1></elmnt1>
16    <elmnt1></elmnt1>
17    <elmnt1></elmnt1>
18    <elmnt1></elmnt1>
19    <elmnt1></elmnt1>
20    <elmnt1></elmnt1>
21    <elmnt1></elmnt1>
22    <elmnt2></elmnt2>
23 </rtelmnt>');
24 END;
25 /
```

PL/SQLプロシージャが正常に完了しました。

別の確認方法(選択肢 C が不正解であること)

- DUAL表からSELECTする

```
1  SELECT XMLTYPE('<?xml version="1.0"?>
2  <!DOCTYPE rtelmnt[
3  <!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
4  <!ELEMENT elmnt1 (#PCDATA)>
5  <!ELEMENT elmnt2 (#PCDATA)>
6  ]>
7  <rtelmnt>
8     <elmnt1></elmnt1>
9     <elmnt2></elmnt2>
10    <elmnt1></elmnt1>
11    <elmnt2></elmnt2>
12 </rtelmnt>') FROM DUAL;
```

```
ERROR:
ORA-31011: XML解析に失敗しました
ORA-19202: XML処理
LPX-00103: Warning: ドキュメント構造がDTDと一致しません。
Error at line 10
中にエラーが発生しました ORA-06512:
"SYS.XMLTYPE", 行310
ORA-06512: 行1
```

- 表にINSERTする

```
1  CREATE TABLE test (xml XMLTYPE);
2  INSERT INTO test VALUES (XMLTYPE('<?xml version="1.0"?>
3  <!DOCTYPE rtelmnt[
4  <!ELEMENT rtelmnt (elmnt1*,elmnt2)?>
5  <!ELEMENT elmnt1 (#PCDATA)>
6  <!ELEMENT elmnt2 (#PCDATA)>
7  ]>
8  <rtelmnt>
9     <elmnt1></elmnt1>
10    <elmnt2></elmnt2>
11    <elmnt1></elmnt1>
12    <elmnt2></elmnt2>
13 </rtelmnt>'));
```

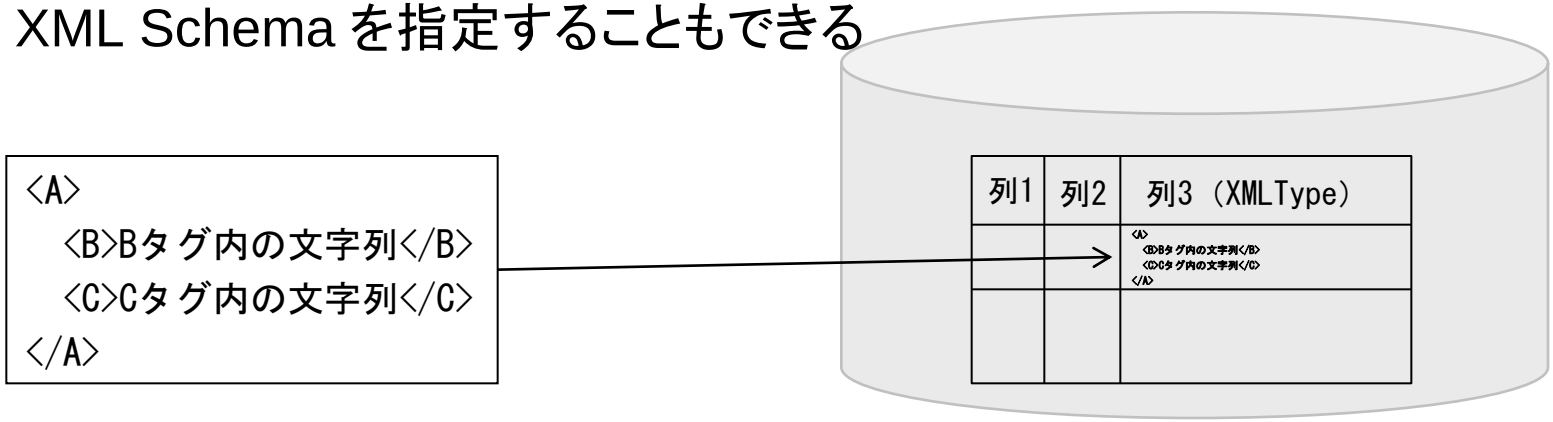
```
INSERT INTO test VALUES (XMLTYPE('<?xml version="1.0"?>
                                 *
行1でエラーが発生しました。:
ORA-31011: XML解析に失敗しました
ORA-19202: XML処理
LPX-00103: Warning: ドキュメント構造がDTDと一致しません。
Error at line 10
中にエラーが発生しました ORA-06512:
"SYS.XMLTYPE", 行310
ORA-06512: 行1
```

XMLType型の物理的な格納方式

- 非構造化記憶域
- バイナリXML記憶域（Oracle Database 11g R1～）
- 構造化記憶域（Oracle9i Database R2～）

非構造化記憶域

- XML文書を、文字データとして物理的に格納する
- 内部的にはCLOB型を利用
  - SecureFiles（データ圧縮、暗号化）を指定可能
- デフォルトの記憶域
- XML Schema の指定は不要
  - 任意の構造を持つXML文書を格納できる
  - XML Schema を指定することもできる



非構造化記憶域(使用例)

- 表作成

```
CREATE TABLE xmltab1 (xml XMLType);
```

- データ格納

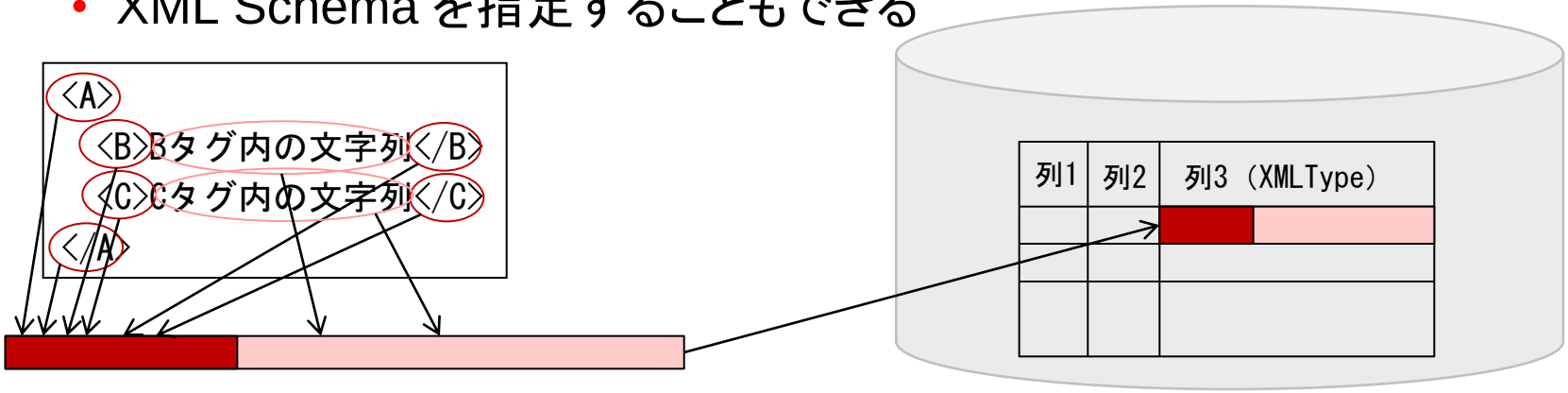
```
INSERT INTO xmltab1 VALUES (XMLType('<A><B>Bタグ内の文字列
</B><C>Cタグ内の文字列</C></A>'));
```

【参考】SecureFiles を指定する場合(推奨)

```
CREATE TABLE xmltab1sf (xml XMLType)
  XMLTYPE COLUMN xml STORE AS SECUREFILE CLOB
  (COMPRESS LOW);
```

バイナリXML記憶域

- XML文書を、XMLの構造部分とコンテンツ部分に分け、独自のバイナリ形式で格納する
- 内部的にはBLOB型を利用
  - SecureFiles（データ圧縮、暗号化）を指定可能
- XML Schema の指定は不要
  - 任意の構造を持つXML文書を格納できる
  - XML Schema を指定することもできる



バイナリXML記憶域(使用例)

- 表作成

```
CREATE TABLE xmltab2 (xml XMLType)
  XMLTYPE COLUMN xml STORE AS BINARY XML;
```

- データ格納

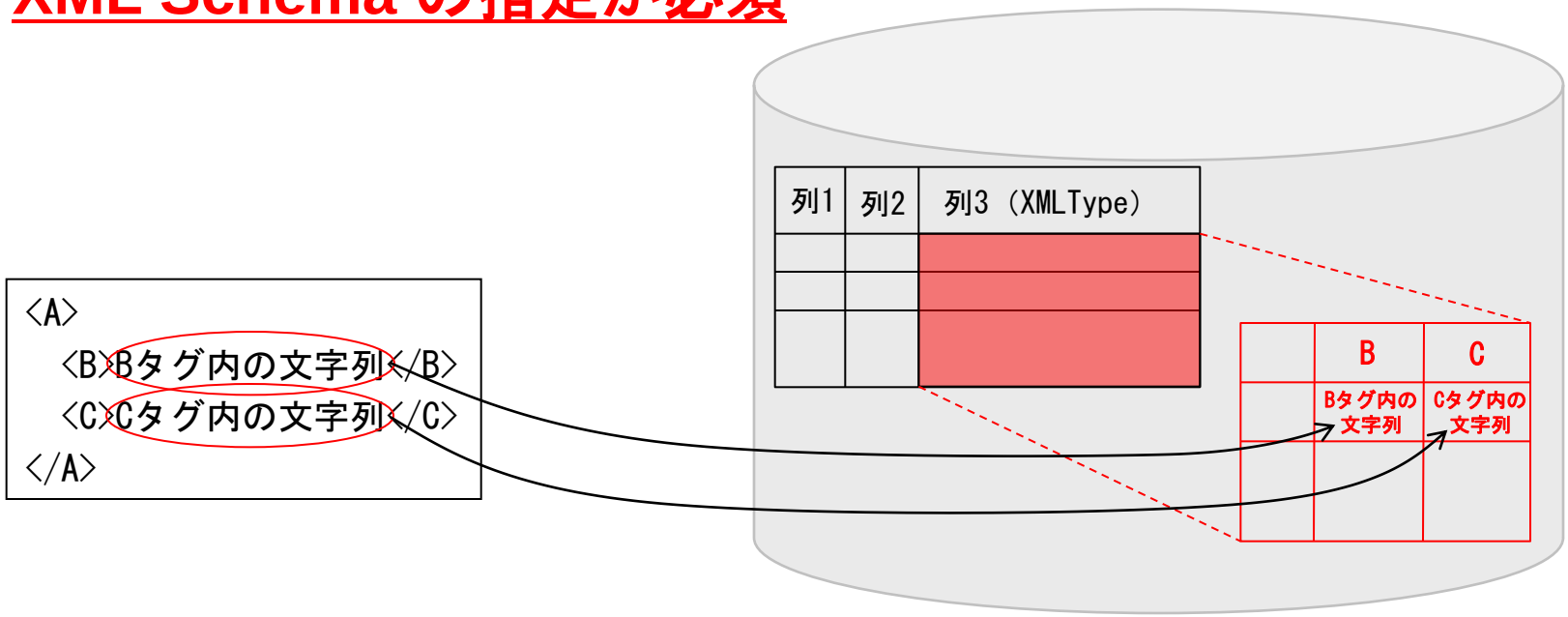
```
INSERT INTO xmltab2 VALUES (XMLType('<A><B>Bタグ内の文字列
</B><C>Cタグ内の文字列</C></A>'));
```

【参考】SecureFiles を指定する場合(推奨)

```
CREATE TABLE xmltab2sf (xml XMLType)
  XMLTYPE COLUMN xml STORE AS SECUREFILE BINARY XML
  (COMPRESS LOW);
```

構造化記憶域

- XML文書を、XML Schema に基づく O/R マッピングによって格納する
- 内部的にはオブジェクト型（O/Rマッピング）を利用
- **XML Schema の指定が必須**



XML Schema

```
<schema
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xdb="http://xmlns.oracle.com/xdb"
    targetNamespace="http://www.oracle.co.jp/a"
    xdb:storeVarrayAsTable="true"
    elementFormDefault="unqualified">
    <element name="A">
      <complexType>
        <sequence>
          <element name="B" type="string"/>
          <element name="C" type="string"/>
        </sequence>
      </complexType>
    </element>
  </schema>
```

← ルート要素

サンプルXML文書

```
<A>
  <B>Bタグ内の文字列</B>
  <C>Cタグ内の文字列</C>
</A>
```

XML Schema のOracle Databaseへの登録

```
1  DECLARE
2    doc VARCHAR2(4000) :=
3    '<schema
4      xmlns="http://www.w3.org/2001/XMLSchema"
5      xmlns:xdb="http://xmlns.oracle.com/xdb"
6      targetNamespace="http://www.oracle.co.jp/a"
7      xdb:storeVarrayAsTable="true"
8      elementFormDefault="unqualified">
9      <element name="A">
10       <complexType>
11         <sequence>
12           <element name="B" type="string"/>
13           <element name="C" type="string"/>
14         </sequence>
15       </complexType>
16     </element>
17   </schema>';
18 BEGIN
19   DBMS_XMLSCHEMA.REGISTERSCHEMA(
20     schemaurl => 'http://www.oracle.co.jp/a.xsd',
21     schemadoc => doc,
22     gentypes  => TRUE,
23     gentables => FALSE
24   );
25 END;
26 /
```

構造化記憶域(使用例)

- 表作成

```
CREATE TABLE xmltab3 (xml XMLType)
  XMLTYPE COLUMN xml STORE AS OBJECT RELATIONAL
  XMLSCHEMA "http://www.oracle.co.jp/a.xsd" ELEMENT "A";
```

- データ格納

```
INSERT INTO xmltab3 VALUES (XMLType('<a:A
xmlns:a="http://www.oracle.co.jp/a"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://www.oracle.co.jp/a http://www.oracle.co.jp/a.xsd">
<B>Bタグ内の文字列</B><C>Cタグ内の文字列</C></a:A>'));
```

XML Schema の登録で Oracle Database に何が起きるのか

```
<element name="A">
  <complexType>
    <sequence>
      <element name="B"
               type="string"/>
      <element name="C"
               type="string"/>
    </sequence>
  </complexType>
</element>
```

```
CREATE OR REPLACE TYPE
"SCOTT"."Entry183_T" AS OBJECT
(
  "SYS_XDBPD$"
   "XDB"."XDB$RAW_LIST_T",
  "B"     VARCHAR2(4000),
  "C"     VARCHAR2(4000)
) FINAL INSTANTIABLE
```

自動的な XML Schema から Oracle データ型（オブジェクト型）への構造のマッピング

データ型のマッピング

スライドには一部のマッピングを掲載しています。

XML Schema	Oracle データ型	
	デフォルト	指定可能
integer	NUMBER	―
float, double	NUMBER	FLOAT DOUBLE BINARY FLOAT
dateTime, time	TIMESTAMP	DATE TIMESTAMP WITH TIMEZONE
boolean	RAW(1)	VARCHAR2
string, ID, IDREF, anyURI, anyType, anySimpleType	VARCHAR2(4000)	CHAR CLOB

XMLType型で提供される記憶域

- XML 文書のデータ特性に合った適切な記憶域を選択可能

構造が決まっている場合

- データのみが意味を持つことが主
- ほとんどタグ構造に変更がない
- 例：従業員レコード（フィールドは従業員番号、氏名、アドレスなど）

→ **構造化記憶域**
メモリー管理の最適化、必要な記憶域の削減
構造は簡単には変更できない

タグ構造が自由で構造が決まっていない場合

- 多数の要素がテキスト・ノードと子要素の両方を含んでいる
- 多数のXML要素が欠落していたり、異なった順序で表示されたりする
- 例：作成者、日付およびタイトルフィールドを持つ書式自由の仕様
- 例：1つのXML Webドキュメント、または書籍のうちの1章

XML 文書全体の検索・更新をする頻度

高い → **非構造化記憶域（CLOB）**
XML文書全体の挿入および取出し時のスループットが構造化記憶域より高い

低い → **バイナリXML記憶域（BLOB）**
問合せのパフォーマンスが非構造化記憶域より高く、XML解析のボトルネックもない
構造化記憶域と同様、ピース単位の更新が可能

データ特性に適した豊富な格納方式

XMLType 型で提供される記憶域の種類と特徴

	構造化記憶域	非構造化記憶域	バイナリXML記憶域
格納効率 （SecureFiles未使用時）	◎	△	○
格納効率 （SecureFiles圧縮使用時）	◎	○	○
XPath ベースの 検索パフォーマンス	◎ ※XPath リライトされ検索	△ ※CLOB データから DOM ツリー構築後に検索	○ ※DOM ツリーは構築されないため、非構造化よりも高速に検索
更新処理パフォーマンス （XML文書全体）	△ ※これは、CPU時間での比較。ディスクI/Oとの相関で、一概に言えない部分もある	◎	○
更新処理 パフォーマンス （XML文書の一部分）	◎ ※部分更新	△ ※全てのデータが更新	○ ※SecureFiles 使用時は部分更新
構造変更の柔軟性	△	◎	◎

Oracle Database で XPath を使う

XPath関連のSQL関数（抜粋）

- **XMLCast** … XMLインスタンスあるいはフラグメントからテキストノードを取り出し、指定されたSQLデータ型へ変換する
- **XMLQuery** … 与えられたXMLインスタンスに対してXPathあるいはXQueryを適用する
- **XMLExists** … 与えられたXMLインスタンスに対してXPathあるいはXQueryを適用し、結果が空でなければTRUEを、結果が空であればFALSEを返す
- **XMLTransform** … 与えられたXMLインスタンスに対してXSLTスタイルシートを適用する
  ※XSLTスタイルシート内でXPathを利用

準備

- XMLType型の列を1つだけ持つ表を作成する

```
CREATE TABLE testxml (xml XMLTYPE)
  XMLTYPE COLUMN xml STORE AS SECUREFILE BINARY XML
  (COMPRESS LOW);
```

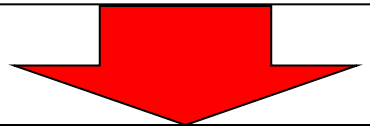
- この表に3件のデータを格納する

```
INSERT INTO testxml VALUES
(XMLTYPE('<a><b>dummy_b1</b><c>dummy_c1</c></a>'));
INSERT INTO testxml VALUES
(XMLTYPE('<a><b>dummy_b2</b><c>dummy_c2</c></a>'));
INSERT INTO testxml VALUES
(XMLTYPE('<a><b>dummy_b3</b><c>dummy_c3</c></a>'));
COMMIT;
```

特定のタグ値を取得する

- 特定タグ値の取得

```
SELECT XMLCast(XMLQuery ('/a/b' PASSING xml RETURNING
CONTENT) AS VARCHAR2(64)) tag_b FROM testxml;
```



- 実行結果

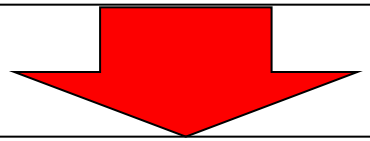
```
TAG_B
--------------------------------------------------------------------

dummy_b1
dummy_b2
dummy_b3
```

特定のタグ値で絞り込み検索を行う(1)

- XMLCast と XMLQuery を利用するパターン

```
SELECT XMLCast(XMLQuery ('/a/b' PASSING xml RETURNING
CONTENT) AS VARCHAR2(64)) tag_b FROM testxml
WHERE XMLCast(XMLQuery ('/a/c' PASSING xml RETURNING
CONTENT) AS VARCHAR2(64))='dummy_c3';
```



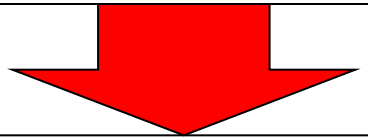
- 実行結果

```
TAG_B
--------------------------------------------------------------------------------
dummy_b3
```

特定のタグ値で絞り込み検索を行う(2)

- XMLExists を利用するパターン

```
SELECT XMLCast(XMLQuery ('/a/b' PASSING xml RETURNING
CONTENT) AS VARCHAR2(64)) tag_b FROM testxml
WHERE XMLExists('/a[c="dummy_c3"]' PASSING xml);
```



- 実行結果

```
TAG_B
--------------------------------------------------------------------------------
dummy_b3
```

XSLTスタイルシートを適用する

```
1  SELECT XMLTransform (xml,'<?xml version="1.0"?>
2  <xsl:stylesheet version="1.0"
3  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4    <xsl:template match="/">
5      <transformed>
6        <xsl:apply-templates select="//b"/>
7      </transformed>
8    </xsl:template>
9    <xsl:template match="b">
10     <xsl:value-of select="../c"/>
11   </xsl:template>
12 </xsl:stylesheet>') result FROM testxml;
```

- 実行結果

```
RESULT
--------------------------------------------------------
<transformed>dummy_c1</transformed>
<transformed>dummy_c2</transformed>
<transformed>dummy_c3</transformed>
```

問題文の引用元
http://www.xmlmaster.org/sample/sample_basic.html#05

サンプル試験問題
XMLマスター:ベーシック 試験(2)

次のXML文書「a.xml」から次の出力結果を生成するXSLTスタイルシート「b.xsl」の(1)に記述できるロケーションパスはどれでしょうか。正しいものを選択してください。

a.xml

```
<?xml version="1.0"?>
<試験>
   <タイトル>XMLマスター試験</タイトル>
   <情報>
     <試験名 試験番号="I10-001">XMLマスター:ベーシック</試験名>
     <試験時間>60分</試験時間>
     <出題数>50</出題数>
     <合格基準>70%以上</合格基準>
   </情報>
   <情報>
     <試験名 試験番号="I10-002">XMLマスター:プロフェッショナル</試験名>
     <試験時間>90分</試験時間>
     <出題数>40</出題数>
     <合格基準>80%以上</合格基準>
   </情報>
</試験>
```

b.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
     <test_info>
       <xsl:apply-templates select="//合格基準"/>
     </test_info>
   </xsl:template>
   <xsl:template match="合格基準">
     <test_no>
       <xsl:value-of select="(1)"/>
     </test_no>
   </xsl:template>
</xsl:stylesheet>
```

出力結果

```
<test_info>
<test_no>I10-001</test_no>
<test_no>I10-002</test_no>
</test_info>
```

A. /試験/情報/試験名[@試験番号]
B. /試験/情報/試験名/@試験番号
C. . . /試験名[@試験番号]
D. . . /試験名/@試験番号

問題文の引用元
http://www.xmlmaster.org/sample/sample_basic.html#05

サンプル試験問題
XMLマスター:ベーシック 試験(2)

次のXML文書「a.xml」から次の出力結果を生成するXSLTスタイルシート「b.xsl」の(1)に記述できるロケーションパスはどれでしょうか。正しいものを選択してください。

a.xml

```
<?xml version="1.0"?>
<試験>
  <タイトル>XMLマスター試験</タイトル>
  <情報>
    <試験名 試験番号="I10-001">XMLマスター:ベーシック</試験名>
    <試験時間>60分</試験時間>
    <出題数>50</出題数>
    <合格基準>70%以上</合格基準>
  </情報>
  <情報>
    <試験名 試験番号="I10-002">XMLマスター:プロフェッショナル</試験名>
    <試験時間>90分</試験時間>
    <出題数>40</出題数>
    <合格基準>80%以上</合格基準>
  </情報>
</試験>
```

b.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <test_info>
      <xsl:apply-templates select="//合格基準"/>
    </test_info>
  </xsl:template>
  <xsl:template match="合格基準">
    <test_no>
      <xsl:value-of select="(1)"/>
    </test_no>
  </xsl:template>
</xsl:stylesheet>
```

出力結果

```
<test_info>
<test_no>I10-001</test_no>
<test_no>I10-002</test_no>
</test_info>
```

A. /試験/情報/試験名[@試験番号]
B. /試験/情報/試験名/@試験番号
C. . . /試験名[@試験番号]
D. . . /試験名/@試験番号

選択肢Dに基づくスタイルシートを、実際に適用してみる

```
SELECT XMLTransform (
  XMLTYPE('<?xml version="1.0"?>
    <試験>
      <タイトル>XMLマスター試験</タイトル>
      <情報>
        <試験名 試験番号="I10-001">XMLマスター:ベーシック</試験名>
        <試験時間>60分</試験時間>
        <出題数>50</出題数>
        <合格基準>70%以上</合格基準>
      </情報>
      <情報>
        <試験名 試験番号="I10-002">XMLマスター:プロフェッショナル</試験名>
        <試験時間>90分</試験時間>
        <出題数>40</出題数>
        <合格基準>80%以上</合格基準>
      </情報>
    </試験>'),
  XMLTYPE('<?xml version="1.0"?>
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:template match="/">
        <test_info>
          <xsl:apply-templates select="//合格基準"/>
        </test_info>
      </xsl:template>
      <xsl:template match="合格基準">
        <test_no>
          <xsl:value-of select="../試験名/@試験番号"/>
        </test_no>
      </xsl:template>
    </xsl:stylesheet>')
) result_d FROM DUAL;
```

- XMLTransform関数を利用

実行結果

```
RESULT_D
--------------------------------
<test_info>
 <test_no>I10-001</test_no>
 <test_no>I10-002</test_no>
</test_info>
```

選択肢A, B, Cに基づくスタイルシートについても、実際に適用してみる

- 選択肢A

```
RESULT_A
------------------------------------------------------
<test_info>
 <test_no>XMLマスター:ベーシック</test_no>
 <test_no>XMLマスター:ベーシック</test_no>
</test_info>
```

- 選択肢B

```
RESULT_B
------------------------------------------------------
<test_info>
 <test_no>I10-001</test_no>
 <test_no>I10-001</test_no>
</test_info>
```

- 選択肢C

```
RESULT_C
------------------------------------------------------
<test_info>
 <test_no>XMLマスター:ベーシック</test_no>
 <test_no>XMLマスター:プロフェッショナル</test_no>
</test_info>
```

Oracle Database で XQuery を使う

XQueryを利用できるSQL関数（抜粋）

- **XMLQuery** ... 与えられたXMLインスタンスに対してXPathあるいはXQueryを適用する
- **XMLTable** ... 与えられたXMLインスタンスに対してXPathあるいはXQueryを適用する（結果をデータベース表として出力）※この項での詳細説明は省略

問題文の引用元
http://www.xmlmaster.org/sample/sample_proDB.html#02

サンプル試験問題
XMLマスター:プロフェッショナル(データベース)試験

次の[example.xml]に対してXQueryによる問い合わせを実行して以下の[出力結果]を得たい。[出力結果]を得ることのできる正しいXQueryをひとつ選択してください。ただし[example.xml]のlog要素のcode属性値は、list要素のcode属性の値と一致するものとなっています。

example.xml

```
1  <logList>
2    <list code="w001" message="警告1"/>
3    <list code="w002" message="警告2"/>
4    <list code="e001" message="エラー1"/>
5    <list code="e002" message="エラー2"/>
6    <day date="2007-12-01">
7      <log time="10:00:00" code="w001"/>
8      <log time="14:00:00" code="e001"/>
9    </day>
10   <day date="2007-12-02">
11     <log time="13:00:00" code="e002"/>
12     <log time="15:00:00" code="e001"/>
13 </day>
14 </logList>
```

出力結果

```
<result>
  <log date="2007-12-01" time="10:00:00" message="警告1"/>
  <log date="2007-12-01" time="14:00:00" message="エラー1"/>
  <log date="2007-12-02" time="13:00:00" message="エラー2"/>
  <log date="2007-12-02" time="15:00:00" message="エラー1"/>
</result>
```

※ 選択肢は次ページに記載

サンプル試験問題
XMLマスター:プロフェッショナル(データベース)試験

問題文の引用元
http://www.xmlmaster.org/sample/sample_proDB.html#02

※ 前頁の問題文に対する選択肢

A.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $log in $doc//log
  return
    <log>{
      $log/../@date,
      $log/@time,
      $doc//list[@code eq $log/@code]/@message
    }</log>
}</result>
```

B.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $log in $doc//log
  return
    <log>{
      ../@date,
      @time,
      ../../list[@code = $log/@code]/@message
    }</log>
}</result>
```

C.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $day in $doc//day
  return
    <log>{
      $day/@date,
      $day/log/@time,
      $doc//list[@code eq $day/log/@code]/@message
    }</log>
}</result>
```

D.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $day in $doc//day
  return
    <log>{
    @date,
      log/@time,
      ../list[@code = $day/log/@code]/@message
    }</log>
}</result>
```

サンプル試験問題
XMLマスター：プロフェッショナル（データベース）試験

問題文の引用元
http://www.xmlmaster.org/sample/sample_proDB.html#02

※ 前頁の問題文に対する選択肢

A.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $log in $doc//log
  return
    <log>{
      $log/../@date,
      $log/@time,
      $doc//list[@code eq $log/@code]/@message
    }</log>
}</result>
```

B.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $log in $doc//log
  return
    <log>{
      ../@date,
      @time,
      ../../list[@code = $log/@code]/@message
    }</log>
}</result>
```

C.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $day in $doc//day
  return
    <log>{
      $day/@date,
      $day/log/@time,
      $doc//list[@code eq $day/log/@code]/@message
    }</log>
}</result>
```

D.

```
<result>{
  let $doc := fn:doc("example.xml")
  for $day in $doc//day
  return
    <log>{
    @date,
      log/@time,
      ../list[@code = $day/log/@code]/@message
    }</log>
}</result>
```

選択肢AのXQueryを実行してみる

```
1  SELECT XMLQuery ('<result>{
2    let $doc := /
3    for $log in $doc//log
4    return
5      <log>{
6        $log/../@date,
7        $log/@time,
8        $doc//list[@code eq $log/@code]/@message
9      }</log>
10 }</result>'
11 PASSING XMLTYPE('<logList>
12   <list code="w001" message="警告1"/>
13   <list code="w002" message="警告2"/>
14   <list code="e001" message="エラー1"/>
15   <list code="e002" message="エラー2"/>
16   <day date="2007-12-01">
17     <log time="10:00:00" code="w001"/>
18     <log time="14:00:00" code="e001"/>
19   </day>
20   <day date="2007-12-02">
21     <log time="13:00:00" code="e002"/>
22     <log time="15:00:00" code="e001"/>
23 </day>
24 </logList>')
25 RETURNING CONTENT) result_a FROM DUAL;
```

- XMLQuery 関数を利用

※ XML文書をSQL関数の引数として渡すため、
let $doc := fn:doc("example.xml")
の行を、
let $doc := /
に書き換えて実行しています。

実行結果

```
RESULT_A
--------------------------------------------------------------------------------
<result>
  <log date="2007-12-01" time="10:00:00" message="警告1"></log>
  <log date="2007-12-01" time="14:00:00" message="エラー1"></log>
  <log date="2007-12-02" time="13:00:00" message="エラー2"></log>
  <log date="2007-12-02" time="15:00:00" message="エラー1"></log>
</result>
```

選択肢B, C, DのXQueryを実行してみる

- 選択肢A以外のXQueryはすべて実行エラーになります
  - 選択肢B

```
PASSING XMLTYPE('<logList>
        *
行10でエラーが発生しました。:
ORA-19277: XPST0005 - XPathステップで、一致するノードがない項目
タイプが指定されます: (parent::node())
```

  - 選択肢C

```
ERROR:
ORA-19121: 重複した属性定義 - time
```

  - 選択肢D

```
PASSING XMLTYPE('<logList>
        *
行10でエラーが発生しました。:
ORA-19277: XPST0005 - XPathステップで、一致するノードがない項目
タイプが指定されます: (parent::node())
```

Oracle Database でリレーショナルデータをXML形式で参照する

リレーショナルデータをXML形式で参照するために利用可能なSQL関数（抜粋）

- **XMLElement** ... リレーショナルデータから単一のXML要素を生成する
- **XMLForest** ... リレーショナルデータから複数のXML要素を生成する
- **XMLAttributes** ... リレーショナルデータからXML要素の属性を生成する

準備

- リレーショナル表を作成する

```
CREATE TABLE testrel
(
  b VARCHAR2(4000),
  c VARCHAR2(4000)
);
```

- この表に3件のデータを格納する

```
INSERT INTO testrel VALUES ('dummy_b1', 'dummy_c1');
INSERT INTO testrel VALUES ('dummy_b2', 'dummy_c2');
INSERT INTO testrel VALUES ('dummy_b3', 'dummy_c3');
COMMIT;
```

「リレーショナル→XML」変換例(1)

- XMLElement を利用

```
SELECT
  XMLElement("a",
    XMLElement("b", b),
    XMLElement("c", c)
  ) result
FROM testrel;
```

- 実行結果

```
RESULT
--------------------------------------------------------------------------------
<a><b>dummy_b1</b><c>dummy_c1</c></a>
<a><b>dummy_b2</b><c>dummy_c2</c></a>
<a><b>dummy_b3</b><c>dummy_c3</c></a>
```

「リレーショナル→XML」変換例(2)

- XMLElement とXMLForestを利用

```
SELECT
  XMLElement("a",
    XMLForest(b AS "b", c AS "c")
  ) result
FROM testrel;
```

- 実行結果

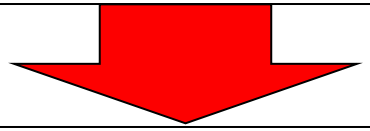
```
RESULT
--------------------------------------------------------------------------------
<a><b>dummy_b1</b><c>dummy_c1</c></a>
<a><b>dummy_b2</b><c>dummy_c2</c></a>
<a><b>dummy_b3</b><c>dummy_c3</c></a>
```

「リレーショナル→XML」変換例(3)

- XMLElement とXMLAttributesを利用(Aタグの属性として列Bを指定)

```
SELECT
  XMLElement("a",
    XMLAttributes(b AS "b"),
    XMLElement("c", c)
  ) result
FROM testrel;
```

- 実行結果



```
RESULT
--------------------------------------------------------------------------------
<a b="dummy_b1"><c>dummy_c1</c></a>
<a b="dummy_b2"><c>dummy_c2</c></a>
<a b="dummy_b3"><c>dummy_c3</c></a>
```

ビューの使用例

- ビューの作成

```
CREATE VIEW xmlview AS
  SELECT
    XMLElement("a",
      XMLElement("b", b),
      XMLElement("c", c)
    ) xml
  FROM testrel;
```

- ビューの参照

```
SQL> SELECT * FROM xmlview;

XML
--------------------------------------------------
<a><b>dummy_b1</b><c>dummy_c1</c></a>
<a><b>dummy_b2</b><c>dummy_c2</c></a>
<a><b>dummy_b3</b><c>dummy_c3</c></a>
```

Oracle Database で XMLデータをリレーショナル形式で参照する

XMLデータをリレーショナル形式で参照するために利用可能なSQL関数（抜粋）

- **XMLCast** … XMLインスタンスあるいはフラグメントからテキストノードを取り出し、指定されたSQLデータ型へ変換する
- **XMLQuery** … 与えられたXMLインスタンスに対してXPathあるいはXQueryを適用する
- **XMLTable** … 与えられたXMLインスタンスに対してXPathあるいはXQueryを適用する（結果をデータベース表として出力）

準備

※ p.28 と同一の内容です。

- XMLType型の列を1つだけ持つ表を作成する

```
CREATE TABLE testxml (xml XMLTYPE)
  XMLTYPE COLUMN xml STORE AS SECUREFILE BINARY XML
  (COMPRESS LOW);
```

- この表に3件のデータを格納する

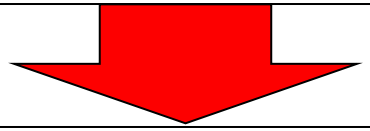
```
INSERT INTO testxml VALUES
(XMLTYPE('<a><b>dummy_b1</b><c>dummy_c1</c></a>'));
INSERT INTO testxml VALUES
(XMLTYPE('<a><b>dummy_b2</b><c>dummy_c2</c></a>'));
INSERT INTO testxml VALUES
(XMLTYPE('<a><b>dummy_b3</b><c>dummy_c3</c></a>'));
COMMIT;
```

「XML→リレーショナル」変換例(1)

※ p.29 の例を2回適用しているだけです。

- XMLCastとXMLQuery を利用

```
SELECT
  XMLCast(XMLQuery ('/a/b' PASSING xml RETURNING CONTENT)
AS VARCHAR2(64)) b,
  XMLCast(XMLQuery ('/a/c' PASSING xml RETURNING CONTENT)
AS VARCHAR2(64)) c
 FROM testxml;
```



- 実行結果

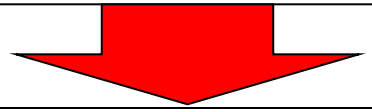
```
B                    C
-------------------- --------------------
dummy_b1             dummy_c1
dummy_b2             dummy_c2
dummy_b3             dummy_c3
```

「XML→リレーショナル」変換例(2)

- XMLTable を利用

```
SELECT x.b, x.c FROM
  testxml t,
  XMLTable('/a' PASSING t.xml
   COLUMNS
    b VARCHAR2(4000) PATH 'b',
    c VARCHAR2(4000) PATH 'c'
  ) x;
```

- 実行結果



```
B                    C
-------------------- --------------------
dummy_b1             dummy_c1
dummy_b2             dummy_c2
dummy_b3             dummy_c3
```

ビューの使用例

- ビューの作成

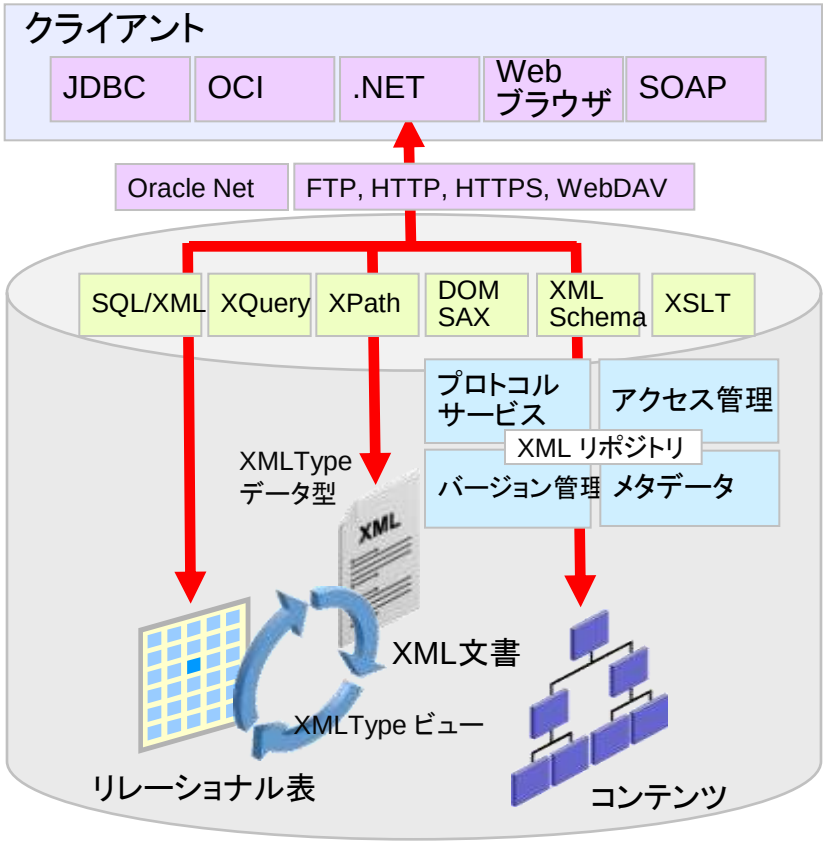
```
CREATE VIEW relview AS
  SELECT
    XMLCast(XMLQuery ('/a/b' PASSING xml RETURNING CONTENT) AS
VARCHAR2(64)) b,
    XMLCast(XMLQuery ('/a/c' PASSING xml RETURNING CONTENT) AS
VARCHAR2(64)) c
   FROM testxml;
```

- ビューの参照

```
SQL> SELECT * FROM relview;

B                    C
-------------------- --------------------
dummy_b1             dummy_c1
dummy_b2             dummy_c2
dummy_b3             dummy_c3
```

Oracle XML DB の全体像

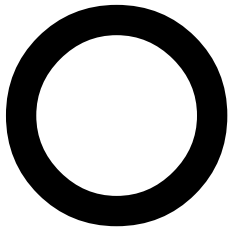


- XML 文書をそのままOracle Database に格納できる
- XML関連の処理は全てデータベース・プロセスによって実行される
- XML 文書の特性に合わせて、効率的な格納方式を選択できる
- 大規模システムへの対応
- 開発生産性
  - W3C 標準規格への準拠
  - XQuery1.0, SQL:2006 (SQL/XML) など技術者の得意な言語で開発ができる
  - 容易な RDBMS データ との連携

"2択"にチャレンジ

問1

- Oracle XML DB は、Enterprise Edition でのみ利用可能であり、利用するには追加のオプション・ライセンスが必要である。



問1（解答と解説）

- Oracle XML DB は、Enterprise Edition でのみ利用可能であり、利用するには追加のオプション・ライセンスが必要である。

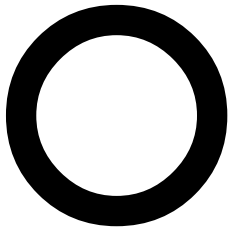
正解は × です。

Oracle XML DB は、Standard Edition One や Standard Edition でも利用することができます。

また、Oracle XML DB を利用するにあたり、追加でオプション・ライセンスを購入する必要はありません。

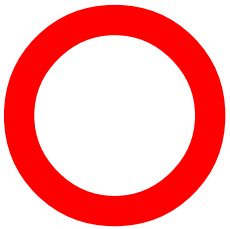
問2

- Oracle XML DB では、リレーショナルデータを、プログラムを介さずにXML形式で参照することができる。



問2（解答と解説）

- Oracle XML DB では、リレーショナルデータを、プログラムを介さずにXML形式で参照することができる。



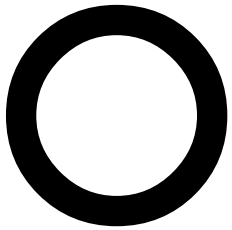
正解は○です。

Oracle XML DB は、SQL/XML標準で規定されたSQL関数を提供しています。したがって、プログラムを介さず、SELECT文のみでリレーショナルデータをXML形式で参照することができます。

(注) PL/SQLはプログラム言語ですが、SQLはプログラム言語ではありません。

問3

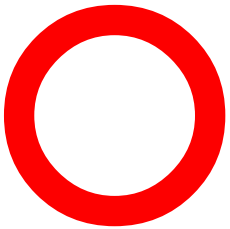
- Oracle XML DB は、DTD と XML Schema の両方に対応している。



×

問3（解答と解説）

- Oracle XML DB は、DTD と XML Schema の両方に対応している。

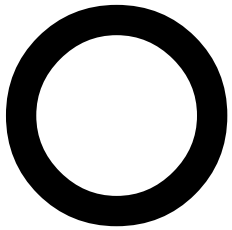


正解は〇です。

XML文書の論理構造を規定するものとしては、DTD と XML Schema がありますが、Oracle XML DB はその両方に対応しています。

問4

- Oracle XML DB のXML処理は、実際にはデータベース外部のプロセスが実行している。



×

問4（解答と解説）

- Oracle XML DB のXML処理は、実際にはデータベース外部のプロセスが実行している。

正解は×です。

Oracle XML DB の構成要素は全て、Oracle Database のデータベース・オブジェクトとして実装されています。このため、XML DB のためのプロセスが個別に起動されることはありません。

ポイント

- Oracle XML DB を使うと、スタイルシート、XPath、XQuery などが簡単に試せる
- データベース層のみでXML関連の全ての処理を実行
- XMLTypeに対するデータ更新・検索のパフォーマンスを出すには、適切な記憶域を選択する必要がある
  - 構造化記憶域を利用した場合、XPath 式の評価、XQuery の実行は、非常に高速（リレーショナル問合せに書き換えられて実行されるため）
- XMLデータとリレーショナルデータの相互変換ができる

参考

- 製品マニュアル
  - 『XML DB開発者ガイド』
    http://download.oracle.com/docs/cd/E16338_01/nav/portal_5.htm#xml
- OTN-J
  - Oracle XML DB
    http://www.oracle.com/technology/global/jp/tech/xml/xmldb/index.html
  - 掲示板（会議室）
    http://otn.oracle.co.jp/forum/index.jspa?categoryID=2
    - 「データベース」カテゴリ→「データベース一般」
    - 「テクノロジー」カテゴリ→「XML」

OTN×ダイセミ　でスキルアップ!!

・一般的な技術問題解決方法などを知りたい！
・セミナ資料など技術コンテンツがほしい！

Oracle Technology Network(OTN)を御活用下さい。

http://otn.oracle.co.jp/forum/index.jspa?categoryID=2

一般的技術問題解決にはOTN掲示版の
「データベース一般」をご活用ください

※OTN掲示版は、基本的にOracleユーザー有志からの回答となるため100%回答があるとは限りません。
ただ、過去の履歴を見ると、質問の大多数に関してなんらかの回答が書き込まれております。

http://www.oracle.com/technology/global/jp/ondemand/otn-seminar/index.html

過去のセミナ資料、動画コンテンツはOTNの
「OTNセミナー　オンデマンド コンテンツ」へ

※ダイセミ事務局にダイセミ資料を請求頂いても、お受けできない可能性がございますので予めご了承ください。
ダイセミ資料はOTNコンテンツ　オン　デマンドか、セミナ実施時間内にダウンロード頂くようお願い致します。

ORACLE

OTNセミナー　オンデマンド　コンテンツ

ダイセミで実施された技術コンテンツを動画で配信中!!

ダイセミのライブ感はそのままに、お好きな時間で受講頂けます。

OTN　オンデマンド　検索

※掲載のコンテンツ内容は予告なく変更になる可能性があります。
　期間限定での配信コンテンツも含まれております。お早めにダウンロード頂くことをお勧めいたします。

オラクル クルクルキャンペーン

あのOracle Database Enterprise Editionが超おトク!!

おトクな買い方
オラクル5年分

- ライセンス使用期間　を5年間に設定
- 初期のライセンスコストがなんと67%OFF　！
- テクニカル・サポート価格も53%OFF　！

Oracle Databaseの
ライセンス価格を大幅に抑えて
ご導入いただけます

多くのお客様でサーバー使用期間とされる
5年間にライセンス期間を限定

- 期間途中で永久ライセンスへ差額移行
- 5年後に新規ライセンスを購入し継続利用
- 5年後に新システムへデータを移行

Enterprise Editionはここが違う!!

- 圧倒的なパフォーマンス!
- データベース管理がカンタン!
- データベースを止めなくていい!
- もちろん障害対策も万全!

この機能でこの価格
ライセンスパック

- Oracle Databaseの機能を存分に使える!
- 2ノードRAC構成も可能!
- サーバー構成によって計4種類のパックから選べる!

詳しくはコチラ
http://www.oracle.co.jp/campaign/kurukuru/index.html

Oracle Direct 0120-155-096

お問い合わせフォーム
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

ORACLE

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。

システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力には、Oracle Direct Seminar申込時と同じ
　ログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録されている連絡先が最新のものになっているか、ご確認下さい。

フリーダイヤル

0120－155－096

※月曜~金曜　9:00~12:00、13:00~18:00
（祝日および年末年始除く）

ORACLE

以上の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。